

Service mesh together with API management

Unifying governance, security, and discoverability across your microservices and APIs

Table of contents

Executive summary

Governance, security, and discoverability are all critical challenges for organizations using or planning to use microservices. You can use service mesh to abstract the governance considerations behind microservices that primarily interact with one another. When combined with API management, you can fully manage, discover and secure all your services — including those that are externally facing.

These considerations regarding microservices are vital. Studies show that 91% of enterprises are using or have plans to use microservices. The benefits are clear — while a monolithic architecture is hard to develop and maintain, a microservices architecture allows for greater agility with its smaller, more targeted services.

However, microservices can't function alone and they often work together to compose larger applications. The integration patterns to accomplish these interactions have evolved from traditional hub-and-spoke enterprise application integration (EAI) to service-oriented architecture (SOA) and, now, to a more API-led approach.

As an organization builds more microservices, complexity grows. The governance and security considerations behind microservice interactions are often custom-coded into the service logic. Teams often build in different languages and deploy to multiple environments, and organizations are typically siloed in the way they manage their services.

In this whitepaper you will learn how you can use a service mesh to solve the governance and security challenges brought by microservices. We will also cover how a service mesh, together with an API management solution, can be used to create a more holistic approach to governing and managing microservices and extend your application network.

01. What microservices got right

Dealing with monolithic applications and architectures can cause development organizations headaches and often creates slower, less-reliable applications and longer development schedules. Companies creating applications in this type of environment will struggle to deliver the experiences their customers expect.

However, this is not inevitable. Microservices are the evolution of best-practice architectural principles. With a microservice-based application architecture, you can build your applications to scale with your business and adapt to customer needs.

Using a microservices approach, large complex applications can be divided up into smaller building blocks of executables that interact to offer the functionality of a highly complex application. To accomplish this, applications are broken down into smaller, independent services with strong network boundaries. These services are not dependent on a specific coding language, so development teams are able to use the language tools they are most comfortable with — accelerating application development.

In Figure 1, you can see a visual comparison between a monolithic and microservices architecture. On the left, you see one large code block that has all the logic associated with delivering a web purchasing experience. On the right, we have smaller services unlocking key pieces of functionality. These services work together to create the desired web experience. Popular microservices patterns have emerged to help customers get the most out of their microservices strategy.

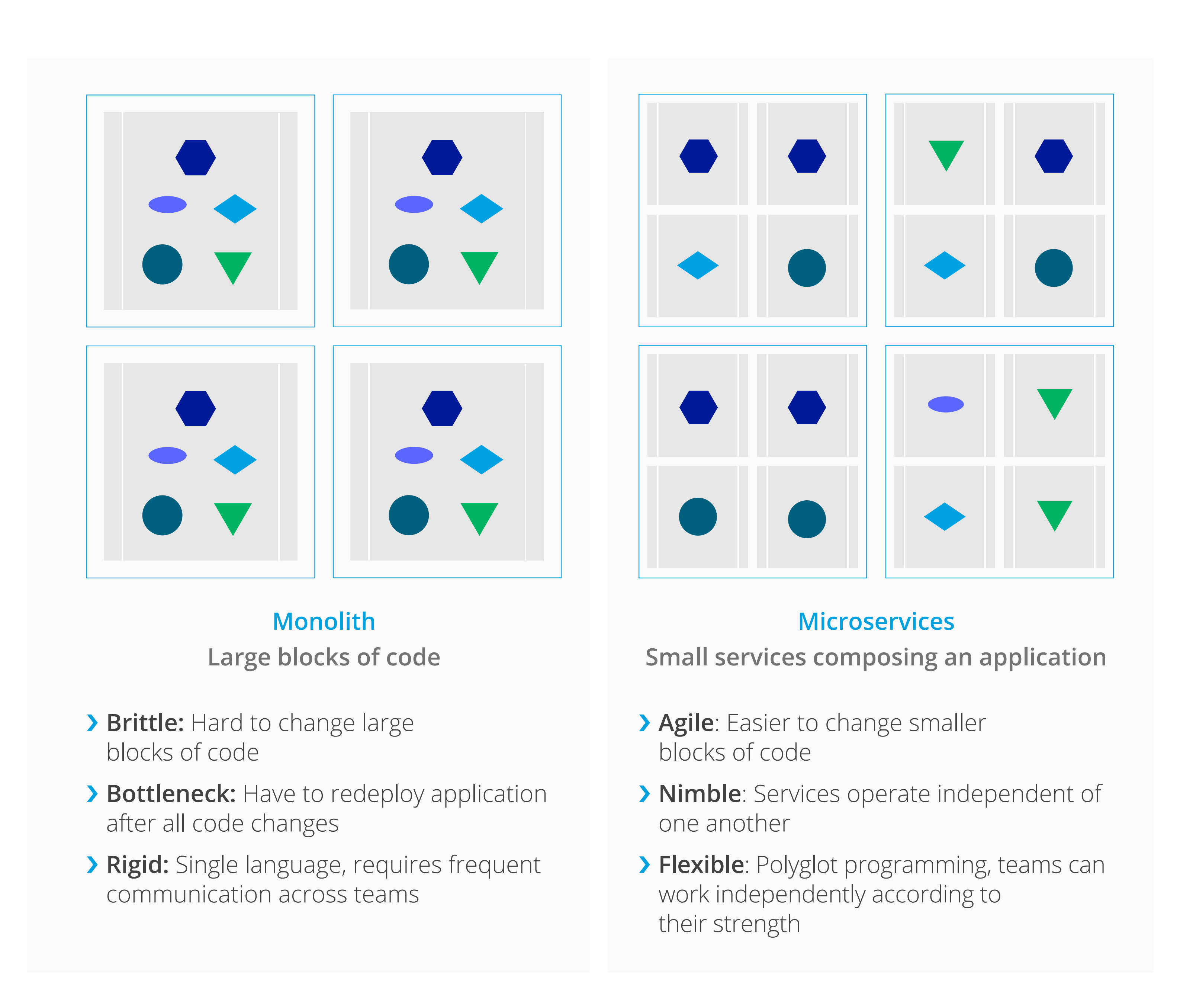


Figure 1: Monolithic vs. microservices architecture.

02. Where microservices alone fall short

Choosing to incorporate microservices as part of your architecture is easy and makes sense. The microservices design pattern for developing software systems has many advantages over developing software in the old monolithic style approach; however, it does come with its challenges.

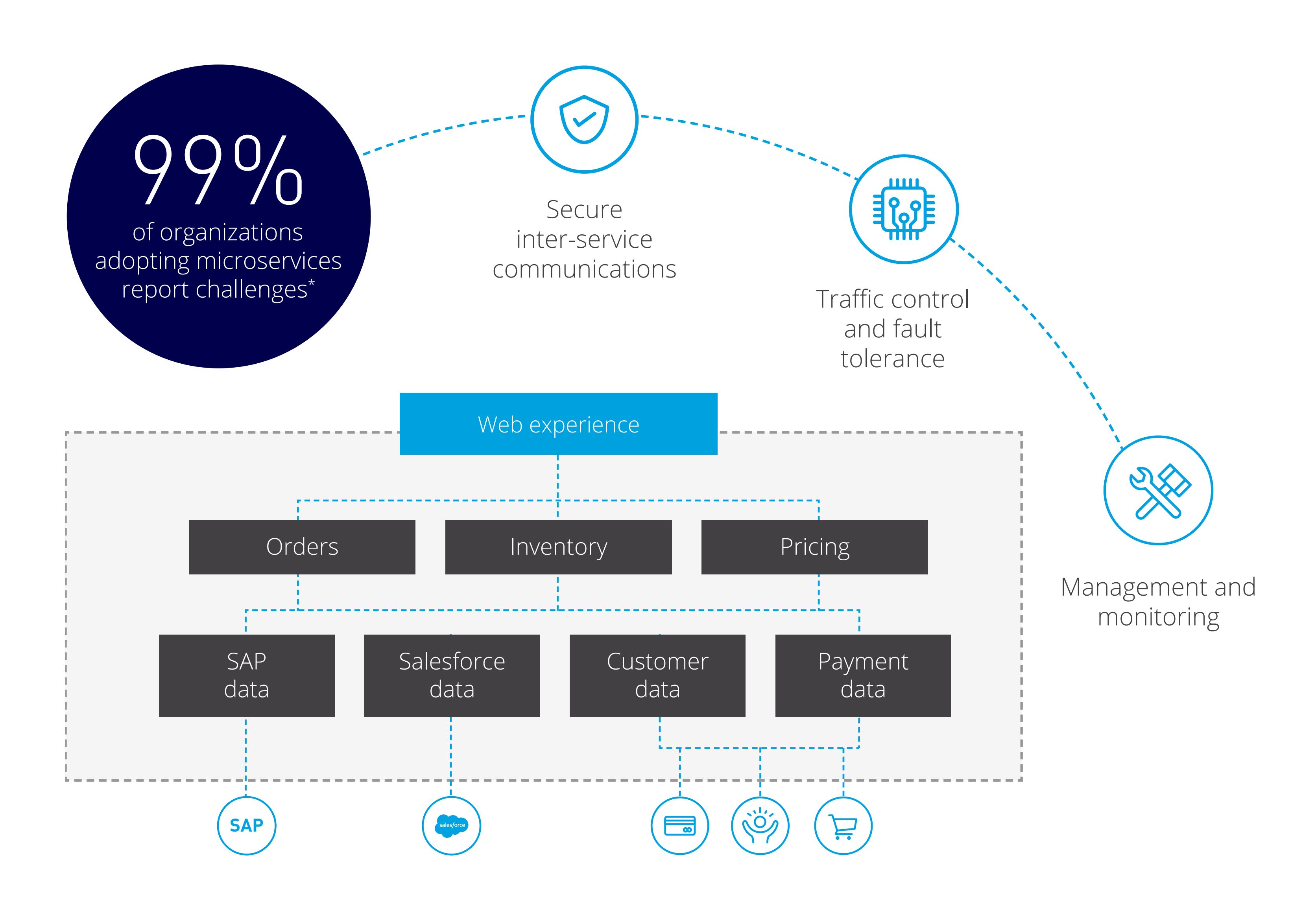


Figure 2: Microservices challenges.

Figure 2 lists some of the main challenges that organizations face when managing and securing their microservices:

1. **Secure inter-service communications:** In order for a microservices-based solution to work, all services need to communicate with each other over network calls. Each of these network calls requires the appropriate level of access, authentication, and authorization (AAA). To further

complicate the situation the AAA needs may differ from one network call to another.

2. **Traffic control and fault tolerance:** A form of traffic management is needed to prioritize all your inter-service network calls. For a variety of reasons, some paths between services might not be available. In these situations, your network must cater for failure situations or fault tolerance.

3. **Management and monitoring:** In a microservices architecture, services are owned and managed by multiple teams. These silos often result in inconsistent policy enforcement and governance. Furthermore, each of these teams might use a disparate set of DevOps tools for management and monitoring.

Martin Fowler, a prominent author and speaker on software development, provides a good comparison of the advantages and challenges that come with adopting microservices. One challenge he highlights is the operational complexity microservices can bring. While each service may be easier to understand compared to a monolith, complexity isn't eliminated.

To solve these challenges, many organizations are forced to custom code governance considerations behind microservices into the service code itself. This complexity can stifle innovation and agility, negating the promise of microservices.

03. How a service mesh solves microservices challenges

A service mesh is an architectural pattern for microservices deployments. Its primary goal is to make service-to-service communications secure, fast, and reliable within a cluster. It also addresses the challenges created by microservices by drawing out common capabilities of security, fault tolerance, and management out of the service code.

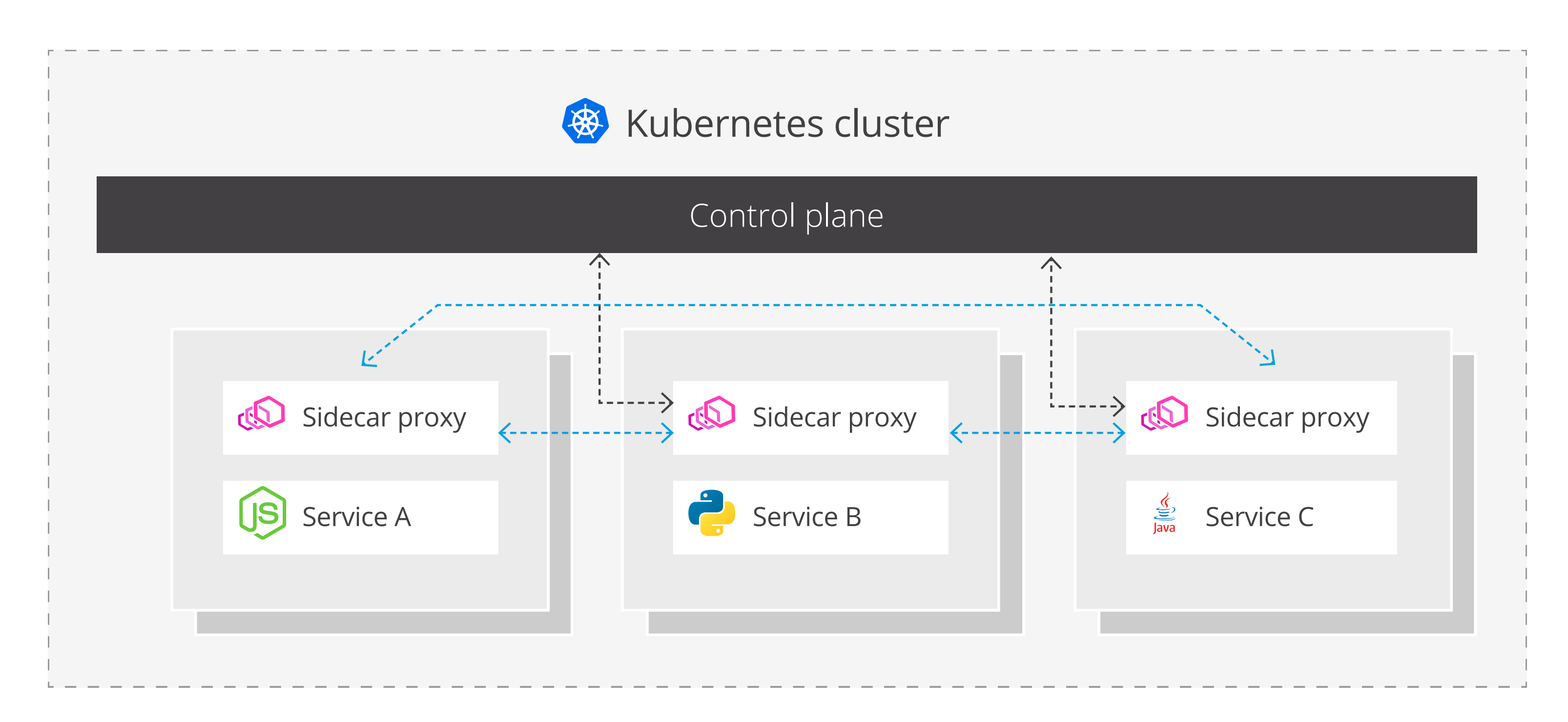


Figure 3: The sidecar pattern for microservices deployments.

There are different patterns to implement these functions — the most common is the sidecar pattern (see Figure 3). Historically, organizations wanting to implement shared functionality had to choose between binding shared libraries into their microservices or inserting a centralized proxy into the architecture. The libraries would become a change management problem, while the proxy could lead to increased latency. A sidecar pattern gives the best of both worlds: your own local proxy that is not bound into your service, giving a cleaner separation and better maintainability over time.

In a sidecar pattern, microservices within a given deployment or cluster interact with each other through sidecar proxy's, or

sidecars. These are lightweight reverse-proxy processes deployed alongside each service process in a separate container.

Sidecars intercept the inbound and outbound traffic of each service, and act according to the security and communication rules as specified by a control plane. The developer can configure and add policies at the control plane level, and abstract the governance considerations behind microservices from the service code, regardless of the technology used to build. Common policies include circuit breakers, timeout implementation, load balancing, service discovery, and security (transport layer security and mutual authentication).

Service mesh as a concept has evolved over time. Initially, microservices adopters embraced the "smart endpoints and dumb pipes" principle for all functionality. They soon realized that it made more sense to provide system-wide policies for common capabilities like routing, rate limiting, and security. Netflix was the first to separate out the application networking layer, creating their famous OSS stack including Hystrix, Eureka, and Ribbon among others. This was followed by Envoy, a high-performance, distributed proxy originally developed at Lyft. Such technologies provided the foundations for the service mesh.

Most service mesh offerings, such as Istio, are deployed into a Kubernetes cluster. While there are many open source service mesh projects and other commercial offerings available, Istio thus far has emerged as the defacto market standard.

04. Is a service mesh enough? What role does API management play?

Service mesh is already viewed as a crucial component in helping solve the operational challenges in managing and governing microservices. According to Gartner and IDC, companies deploying microservices to production will require some form of service mesh capabilities in order to scale.

However, as mentioned before, a service mesh is typically applied to a specific Kubernetes cluster. A service mesh intends to make service-to-service communication reliable, fast, and secure within this cluster. In the enterprise, several services will also have consumers outside any one specific domain — such as developers, web apps, mobile devices, and more.

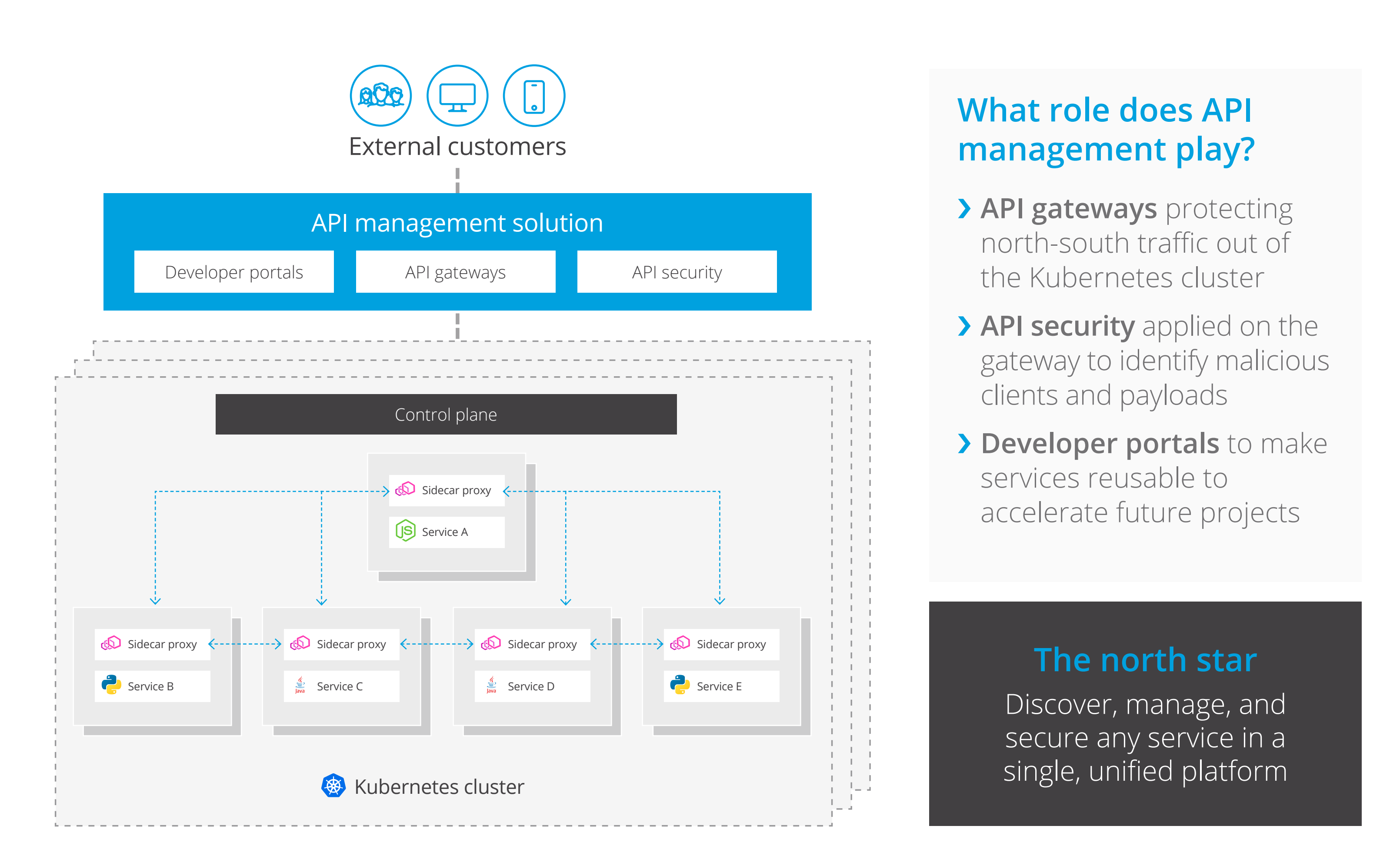


Figure 4: The role of API management.

This is where an API management solution comes into play, as organizations use full lifecycle API management to protect traffic from external consumers. Why an API management solution?

For security reasons, it is best practice to apply an API gateway in front of services with external consumers. Without any edge- or gateway-level security, a malicious user that directly calls a service within a Kubernetes cluster can easily obtain the identity of the cluster namespace and service. For example, a web user can identify this information using the "inspect element" feature on a web browser.

We can then apply API-level security on these gateways. We might apply an OAuth policy to enforce external facing user authentication, XML/JSON threat protection to validate potential malicious payloads, or policies to protect the cluster from DoS and WaF attacks.

Applying the appropriate protections to external facing services opens the door to making these services discoverable. Developers can reuse microservices to avoid any duplication of business logic across multiple services. Ideally, this involves a centralized and secure location through which developers can publish and discover services to use in future projects.

Let's look at an example of a bank to illustrate this. A bank typically has several departments (e.g. mortgage, personal banking, corporate banking, insurance), but has similar needs for services in each department (e.g. credit checks, bank physical locations, etc.). The bank doesn't want to duplicate effort just because these services are used by different departments or exposed to different channels. Ideally, this bank, and any organization using microservices, will have a developer portal or some location where they can publish and reuse these services.

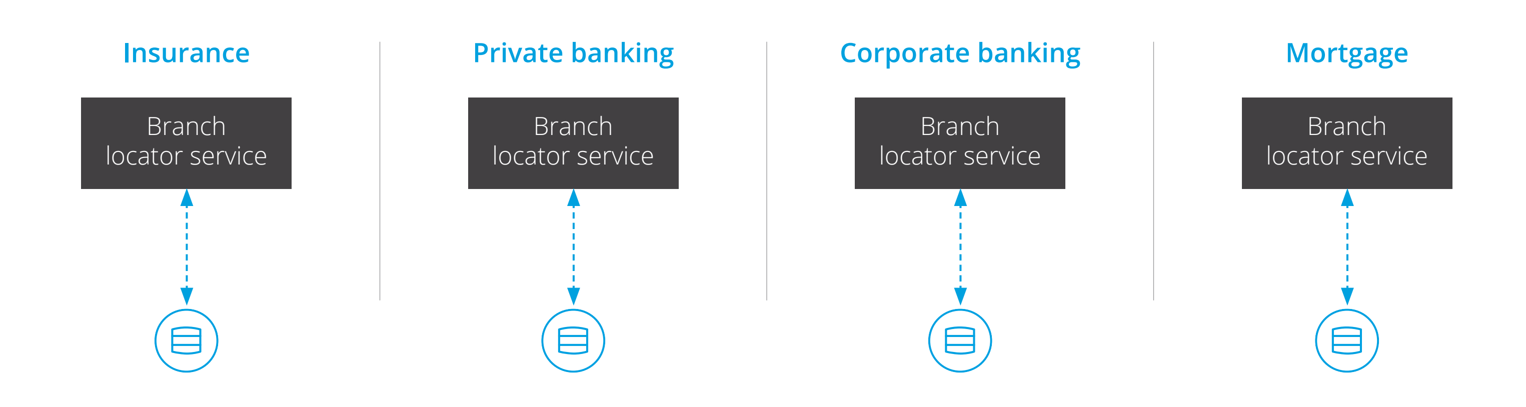


Figure 5: Similar needs across different channels.

For most organizations, it will take years to break the monolith components that currently exist in their software landscape into microservices. It's likely that both monoliths and microservices will coexist for some time. Remaining monoliths still need to be managed and secured, along with all other microservices in an enterprise.

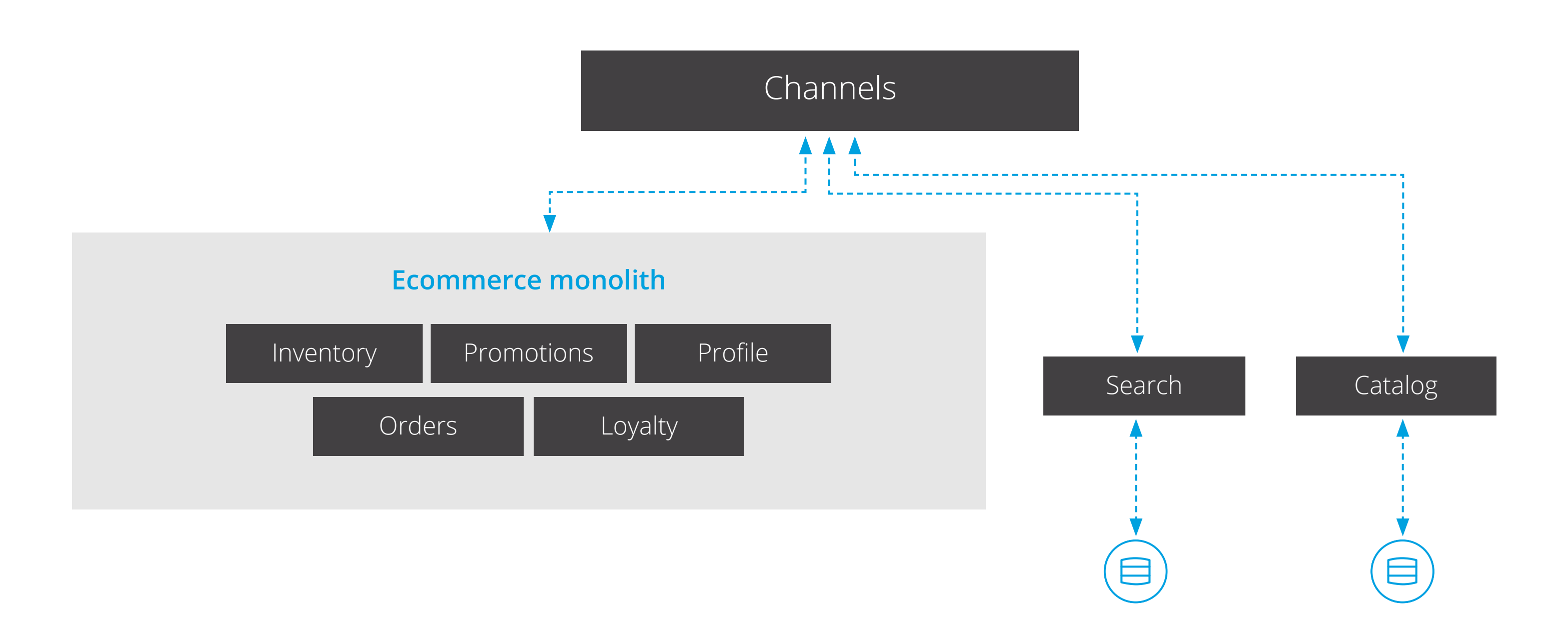


Figure 6: Banks serve their various channels using both monoliths and microservices.

In the complimentary world of API management and service mesh, the need for enterprise-wide governance across the various existing monoliths, SaaS apps, microservices, and other legacy systems is ever increasing. Organizations often invest in multiple solutions and countless resources to meet their needs. It can become incredibly complex to manage.

In the most ideal case, an organization is able to discover, manage, and secure any service in a single, unified platform.

05. Building an application network on MuleSoft's Anypoint Platform

MuleSoft's Anypoint Platform

- **Connect any app, data, or device:** Use prebuilt connectors, templates, and drag-and-drop tools to integrate anything
- **Enable self-service with APIs**: Securely unlock data and drive adoption at scale through a central marketplace
- **Simplify manageability**: Gain insights with the application network graph to ease multi-cloud operations

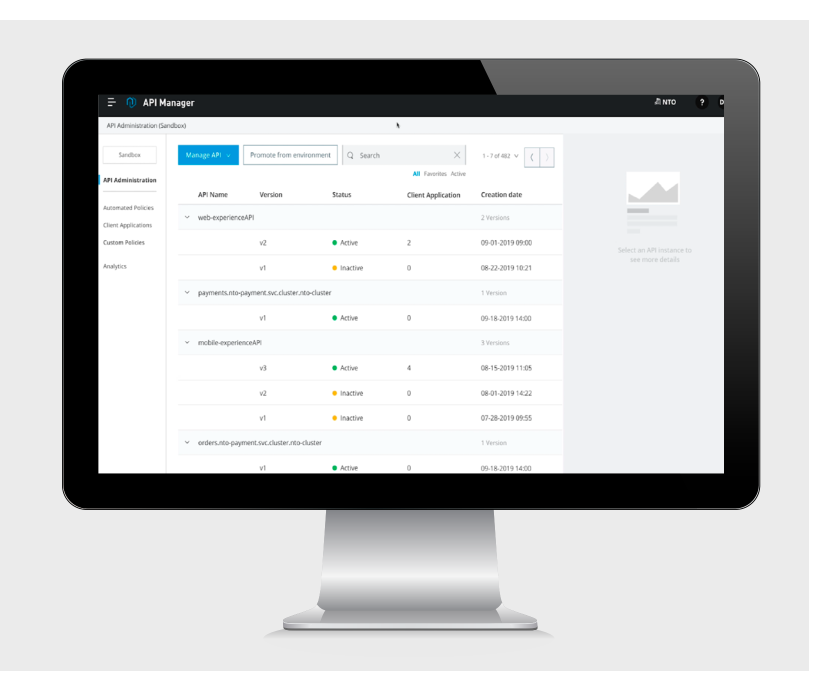


Figure 7: MuleSoft's Anypoint Platform

Anypoint Platform™ is the #1 platform for integrations and APIs. It is used widely to implement a variety of integration use cases like microservices, SaaS integration, API Management, SOA, ESB, and more.

With Anypoint Platform, you can empower your business to:

- Connect any system, app, or data source using prebuilt connectors, templates and drag-and-drop tools to integrate anything.
- Securely unlock your data, wherever it resides, with APIs, and enable teams to self-serve at scale through a central marketplace.

MuleSoft customers can realize a microservices architecture using Anypoint Platform and an API-led approach. These microservices leverage the Mule runtime engine, and become a part of the customers application network — a way to connect applications, data, and devices through APIs.

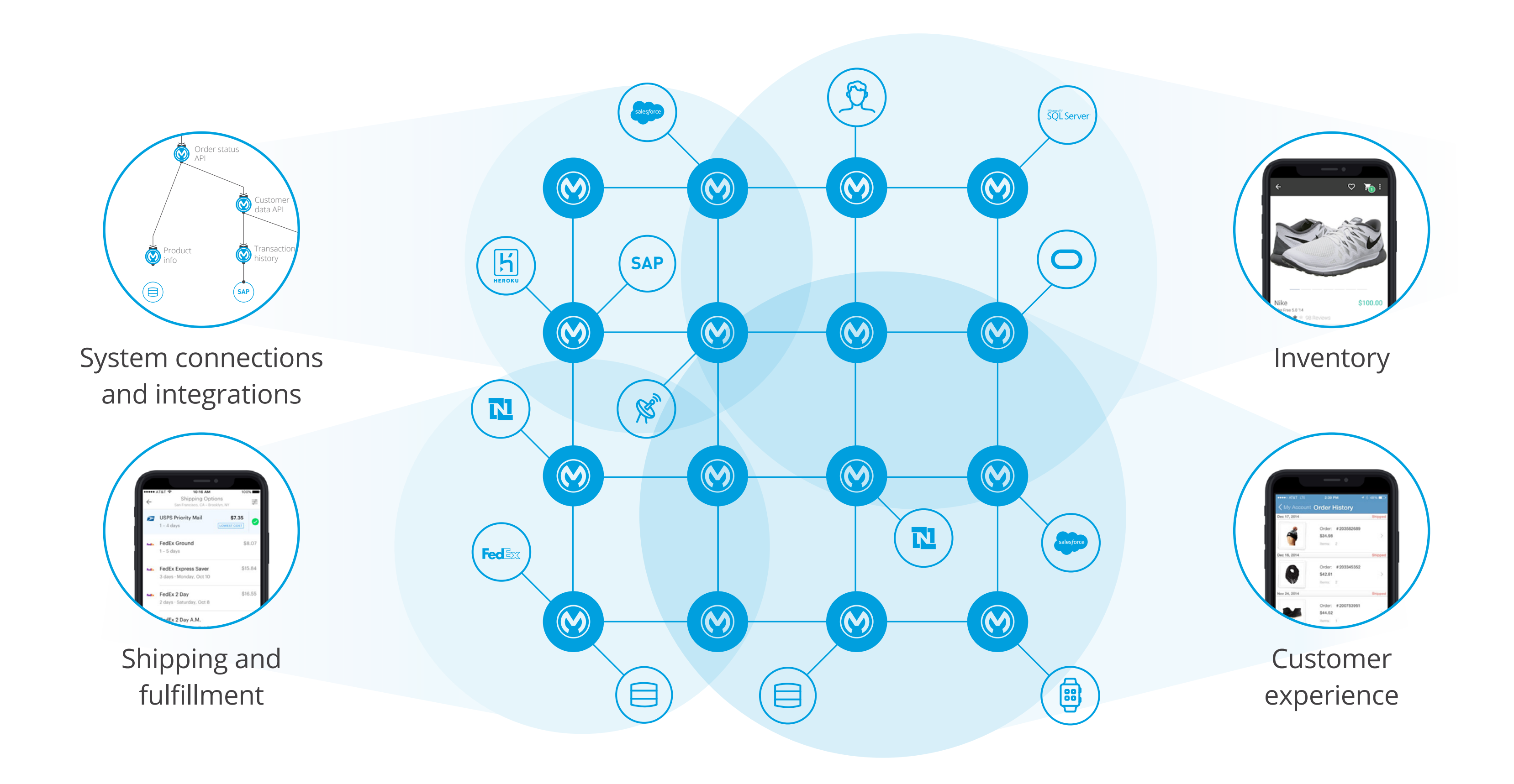


Figure 8: An application network emerges

Anypoint Platform already brings full lifecycle API and service-management capabilities to support your application network.

06. Anypoint Service Mesh — Extending the application network

We recognize, however, that microservices exist outside the MuleSoft ecosystem. That is where Anypoint Service Mesh comes into play. Anypoint Service Mesh is a MuleSoft solution that allows you to discover, manage, and secure any service deployed to Kubernetes — no matter what language it is built in.

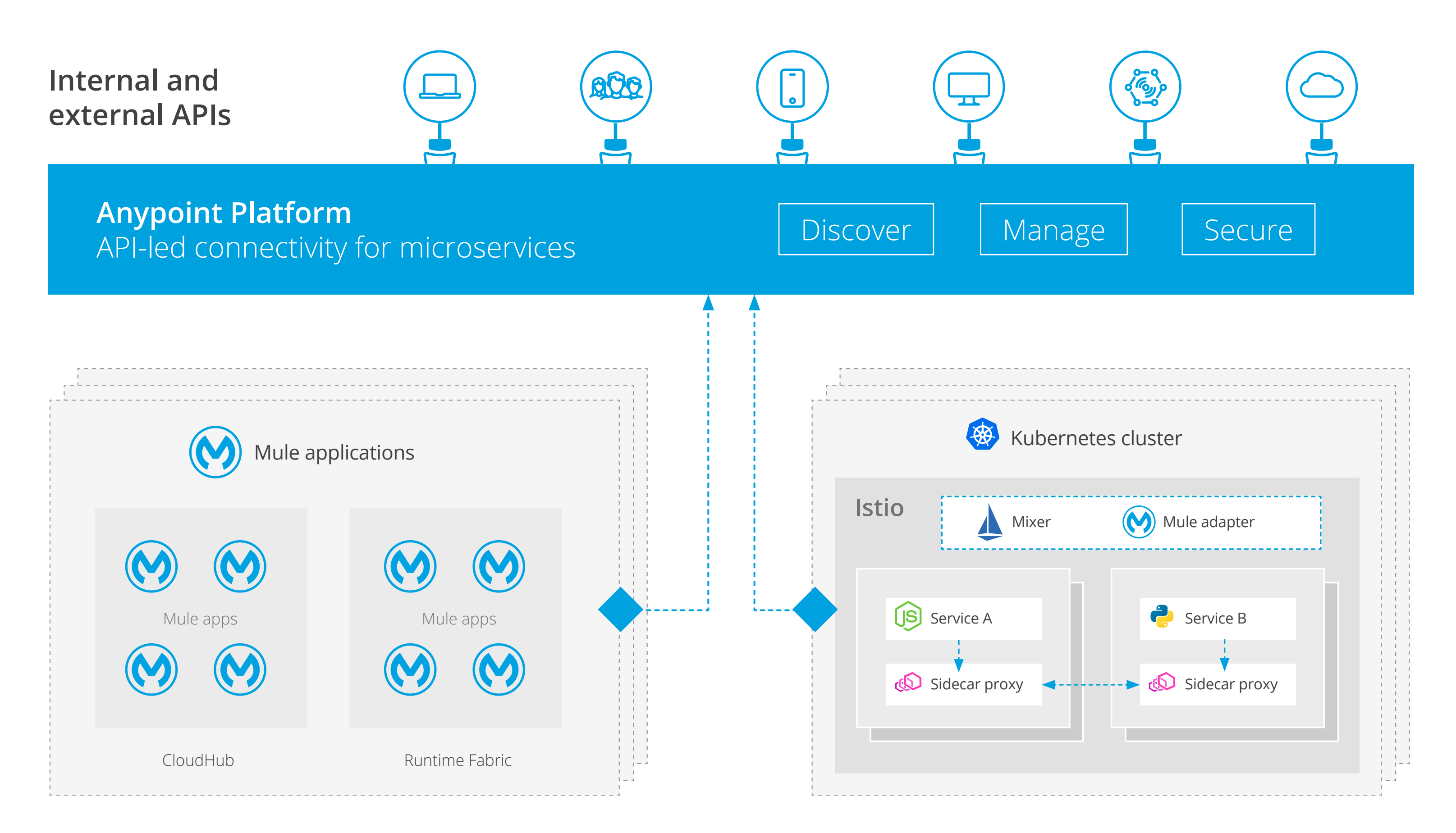


Figure 9: Extend the benefits of an application network with Anypoint Service Mesh.

To do that, we work with the customer instance of Istio, a leading open source service mesh technology, deployed on their kubernetes cluster. We then use a MuleSoft adapter to discover services within the cluster into Anypoint Platform.

With Anypoint Service Mesh, you are able to extend the benefits of the application network to any service, not just

those built leveraging the Mule runtime engine. This allows customers to:

- **Discover and leverage any service in any architecture**
  - Visualize microservice dependencies using the application network graph.
  - Empower innovation teams to build with technologies that best align to their skillsets.
  - Maximize adoption and reuse by adding microservices to Anypoint Exchange.
- **Centrally manage and scale**
  - Ensure resiliency across services with Istio traffic control policies.
  - Measure and optimize performance across all microservices with API analytics.
  - Integrate with existing continuous delivery or CI/CD pipelines.
- **Enable security by default**
  - Ensure zero-trust with Istio and Envoy authentication and authorization policies.
  - Add additional layers of security for consumer facing services.

In this way, we are extending the power of our API management capabilities to all services. With Anypoint Service Mesh, you have the ability to secure your east-west[1] and north-south[2] traffic from a single control plane.

1 East/West traffic is the traffic within a data center network
2 North/South traffic is the traffic coming in and out of a data center network

Conclusion

Microservices are an important piece to making an enterprise more agile and flexible. However, organizations that rely on microservices alone, will have difficulty managing the security and governance complexity that this type of architecture brings. Combine that with teams building in different languages and deploying to multiple environments and organizations will find themselves siloed in how they manage their applications and services. A service mesh can solve many of the security and governance challenges, but when you bring service mesh and API management together into a single platform — such as Anypoint Platform — your organization can centrally manage and scale microservices, enable security by default, and allow developers to discover any service in any architecture.

Watch our demo to learn how a service mesh fits into your microservices strategy and how Anypoint Service Mesh can help you extend your application network to any service.

About MuleSoft

MuleSoft, a Salesforce company

MuleSoft, the world's #1 integration and API platform, makes it easy to connect data from any system — no matter where it resides — to create connected experiences, faster. Thousands of organizations across industries rely on MuleSoft to realize speed, agility and innovation at scale. By integrating systems and unifying data with reusable APIs, businesses can easily compose connected experiences while maintaining security and control. Through API-led connectivity, customers unlock business capabilities to build application networks that deliver exponentially increasing value. MuleSoft is the only unified platform for enterprise iPaaS and full lifecycle API management, and can be deployed to any cloud or on-premises with a single runtime.

For more information, visit **mulesoft.com**